DOCUMENT RESUME

ED 333 863 IR 015 036

AUTHOR Oulman, Charles S.; Lee, Motoko Y.
TITLE Monte Carlo Simulation Using HyperCard and Lotus 1-2-3.
PUB DATE 90
NOTE 14p.; In: Proceedings of the Annual Academic Microcomputing Conference (6th, Columbus, OH, April 22-25, 1990) p.231-243.
PUB TYPE Guides - Non-Classroom Use (055) -- Computer Programs (101) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programing); Computer Assisted Instruction; *Computer Simulation; *Computer Software; Higher Education; Hypermedia; Microcomputers; *Monte Carlo Methods; Spreadsheets
IDENTIFIERS *HyperCard; *LOTUS 1 2 3

ABSTRACT

Monte Carlo simulation is a computer modeling procedure for mimicking observations on a random variable. A random number generator is used in generating the outcome for the events that are being modeled. The simulation can be used to obtain results that otherwise require extensive testing or complicated computations. This paper describes how Monte Carlo simulation can be done using LOTUS 1-2-3 and HyperCard for use as an instructional tool in the classroom. Ten figures illustrate various aspects of the simulation. (Author/BBM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED333863

Sixth Annual Academic Microcomputing Conference
Columbus, Ohio
April 22—25, 1990

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Monte Carlo Simulation Using HyperCard and Lotus 1-2-3

Charles S. Oulman and Motoko Y. Lee
Iowa State University

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Charles S. Oulman

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

1201S036

Monte Carlo Simulation Using HyperCard and LOTUS 1-2-3

Charles S. Oulman, Professor, Dept. of Civil & Construction Engineering,
Motoko Y. Lee, Associate Professor, Dept. of Sociology & Anthropology

Iowa State University, Ames, Iowa 50011
(515) 294-6734 & 8440, S2.MYL@ISUMVS.BITNET

Monte Carlo simulation is a computer modeling procedure for mimicking observations on a random variable. A random number generator is used in generating the outcome for the events that are being modeled. The simulation can be used to obtain results that would otherwise require extensive testing or complicated computations. We will show how Monte Carlo simulation can be done using LOTUS 1-2-3 and HyperCard for use as an instructional tool in the classroom.

INTRODUCTION

Simulation models are used to predict the behavior of systems. In simple systems, the system output may be adequately predicted by observing the effects of changes in a single system variable. In complex systems, the output might depend upon the interaction of a large number of components.

Monte Carlo simulations have been used in analyzing complex systems since the 1940s. Von Neumann and Ulam, two mathematicians working in nuclear physics, used Monte Carlo simulations to solve a problem that was too expensive to solve experimentally and too difficult to solve analytically.

In a Monte Carlo simulation, a computer generated random number is used as the argument in a function to determine the outcome of an event. Computers are used to keep track of the outcomes and to analyze the results of a series of trials.

Four examples will be used to demonstrate how LOTUS 1-2-3 or HyperCard can be used as a programming language for Monte Carlo simulations. Three of the examples are based on examples and problems from Chapter 3 of our book: Macro Programming Using 1-2-3 for Engineers (Oulman and Lee, 1990). A fourth example, the single server simulation, is based on an example from Chapter 17 of the third edition of Operations Research (Taha, 1982).

LOTUS SHELL

Performing a Monte Carlo simulation with LOTUS 1-2-3 is simplified by using a macro-driven shell to do that. Figure 1 shows the basic part of the shell, i.e. the two macros that we have used for this purpose. The first is a labeling macro (L-macro) for labeling the cells that will store variables in the simulation. The second is an execution macro (X-macro) for executing the simulation. These two macro programs will be used for all four simulations. For each simulation, additional program lines are needed to perform the computations that are unique to the specific simulation being executed.

The L-macro program (the first part of Figure 1) places five labels in a column of consecutive cells and name the cells to the right with those labels (rnlr for Range Name Label Right). It then skips a row and types another label and names the cell on its right with that label. The named cells perform the following functions during the simulation:

NUMBER	stores the random number,
COUNT	stores the trial turn in progress,
LIMIT	contains the maximum number of trials to be executed,
SCRATCH	is a scratch pad for intermediate computations and other information that must be processed,
TOTAL	stores the total of the scores during the simulation and finally a mean score for the simulation (total score/limit), the mean score for the set of trials.
START	contains the first line of the program (a subroutine) that is unique to the specific simulation.

To create a worksheet for a simulation, the L-macro is invoked in a cell such as Cell A23, the one used as the starting point in the first example.

The X-macro erases the previous data in SCRATCH, TOTAL, and COUNT to get ready for the new run. It moves the cursor to SCRATCH and calls the subroutine that begins in START until the designated limit is reached. (Note: the operation of the {for} statement ends when the value in COUNT becomes LIMIT+1. The LIMIT+1th trial is not included in the computation.) After exiting from the FOR-loop, the X-macro writes the mean value (TOTAL/LIMIT) into TOTAL. The L-macro and the X-macro are used in each of our simulations.

USING THE LOTUS SHELL

The programming lines beginning in START are unique to a specific simulation. Figure 2 shows the work area for the first example. Only one program line is needed in this subroutine. The simulation calculates the area under the curve for the equation y = exp(x) over the range in x-values from 0 to 1. We can estimate the area by multiplying the average value of y by the range in x-values. Since the random number function returns values between 0 and 1, the x-value for a trial is simply the random number and the area is the sum of the y-values divided by the number of trials. The program line in START contains a {calc} that calculates a new random number to put in NUMBER and a LET-command which types the sum of the new y-values and TOTAL into TOTAL.

Figure 3 shows the work area for our second example. This simulation estimates the average life of a system containing components connected in series, such as light bulbs or electrical relays. Each component has a life (t) measured in years with a cumulative distribution function (P) specified by:

$$P = 1 - \exp(-t)$$

Thus

$$t = \ln(1 - P)$$

If either component in a series fails, the whole system fails. In this simulation, the life of each component is calculated for a given trial and the one smaller of the two values is used as the life of the system. (Note: the cursor remains in SCRATCH throughout the simulation.)

For a given trial, two random numbers are generated and the life corresponding to each random number is calculated. The life corresponding to the first random number is stored in LIFE and the life corresponding to the other is stored in SCRATCH. Next, the value in SCRATCH is compared with the value in LIFE, and if the one in SCRATCH is shorter, it is used to replace the one in LIFE. Finally, the value of TOTAL is added to the value in LIFE and the sum is used to update the value in TOTAL. After all the trials have been completed, the value in TOTAL is divided by the value in LIMIT and the quotient is stored in TOTAL as an estimate of the average life of the system.

Figure 4 shows the work area for the third example. The simulation is used to determine the steady state temperature at a specified location within an infinitely long bar with a rectangular cross-section — a 3 cm x 3 cm square in the example. One side is maintained at a constant temperature of 100°C and the remaining sides are maintained at 20°C.

The cross section of the bar will be modeled as an array of square finite elements with the centers of the exterior elements located on the walls of the bar. The temperature of each exterior element (excepting the corner elements) is the same as the temperature of the wall on which it is centered. The temperature of an interior element can be approximated as the average of the temperatures of the four elements sharing a common side.

If the bar in our example is modeled with a 4 x 4 grid, the temperature at an interior cell can generally be found by solving 4 simultaneous equations. (Implicitly, the elements are 1 cm x 1 cm.) Because of the bilateral symmetry for the specified boundary conditions in our example, the temperature can be found by solving 2 simultaneous equations. The temperature at the two points that are 1 cm from each of two walls maintained at 20°C will be 30°C and the temperature at the other two points will be 50°C.

The solution for the temperature of a specified element can also be approximated with a "random walk" starting at the specified point and ending at one of the walls. In a random walk, the temperature at the wall where the walk terminates is recorded. The recorded temperature represents a random variable because it is uniquely determined by the outcome of a random experiment. The random variable possesses the property that its mean is equal to the temperature corresponding to the starting point in the random walk.

A coordinate system has been used to specify locations in the bar. The 100°C face is located at a y-value of 3. The random walk in our example will start at x and y coordinates of 1,1 and end when the value of either coordinate becomes 0 or 3. The first line in the program initializes the x and y coordinates to values of 1 and 1. The next three lines in the program used in the random walk determine the new coordinates for a step in the walk. We assume that the probability of moving to any of the four surround cells is 0.25. The first line will converts a random number to an integer with a value of 0,1,2, or 3.

The second line in the program subtracts 1 from the x coordinate if the integer is 0, adds 1 to the x coordinate if the integer is 1, and leaves the x coordinate unchanged if the integer is 2 or 3. The third line in the program subtracts 1 from the y coordinate if the integer is 2, adds 1 to the y coordinate if the integer is 3, and leaves the y coordinate unchanged if the integer is 0 or 1.

BEST COPY AVAILABLE

The next four lines in the program determine when the walk has ended and assign a temperature based on the temperature of the wall that is reached. The walk ends and a temperature of 20°C is added to the value in TOTAL when the x coordinate becomes 0 or 3 or the y coordinate becomes 0. The walk ends and a temperature of 100°C is added to the value in TOTAL when the x coordinate becomes 3. If a given step in the walk does not end at a wall, a new random number is generated and the trial proceeds with another step in the random walk.

Figure 5 shows the work area for the last example, a single-server simulation applied to advising undergraduate students. The arrival times for students arriving at the advisors office has been modeled with a Poisson distribution with the mean arrival time being specified for a given simulation. The time spent advising a given student depends on the students classification and the nature of the help requested from the advisor.

For this simulation, the percentage of freshmen, sophomores, juniors and seniors has been assumed to be 35, 30, 25 and 20 percent respectively. The probability that the help requested will be a "drop and add", scheduling of classes or a more complex advising matter differs with the students classification. The length of time spent with the student depends on both classification and the nature of the help requested.

Thus, the arrival time and duration of of the current advising activity for each trial in the simulation has been based on three random numbers. The first line in the program generates the first random number, updates the arrival time of the student and stores the arrival time in ARR. The second line determines whether the arrival is before or after the time that the previous student will leave the advisor's office. If the current student arrives before the previous student leaves, the difference between the leaving and the arriving time is added to the accumulated waiting time.

The third line in the program generates the second random number and uses the random number as the argument in a lookup function to obtain a classification for the current student from a lookup table. The lookup tables are shown in Figure 6.

The remaining lines in the program generates the third random number and uses that random number as the argument in another lookup function to obtain the expected duration of the advising activity and the expected departure time. If the current student arrives before the previous student leaves, the duration is added to the previous student's leaving time.

HYPERCARD SHELL

Monte Carlo simulations can be performed in a HyperCard shell consisting of a card with several fields and buttons. Figure 7 shows the card at the end of the Single Server for Advising simulation (the fourth example) for a mean arrival time of 15 minutes and 20 trials. Figure 8 shows the fields that are visible during the simulation. Figure 9 shows the lookup table field that contains the classification and duration values used by the lookup function. Figure 10 shows the field containing the program used for the specific simulation.

The script for the DO IT button is similar in form and function to the X-macro used in the 1-2-3 shell:

```
on mouseUp
  repeat with i = 1 to 5
```

```
    put 0 into line i of card field "OUTPUT"
  end repeat
  put card field "LIMIT" into m
  repeat with i = 1 to m
    put i into line 1 of card field "OUTPUT"
    put false into done
    put 0 into j
    repeat until done
      put j + 1 into j
      get line j of card field "BOX"
      if it is not empty then do line j of card field "BOX"
      if it is empty then put true into done
    end repeat
  end repeat
  put value of ((line 2 of card field "OUTPUT")/m) into ¬
  line 2 of card field "OUTPUT"
end mouseUp
```

LIMIT is a one-line field containing the limiting value for the number of trials in the simulation. OUTPUT is the five-line field used in displaying values generated in each trial. BOX is a scrolling field that contains the program used to run the specific simulation.

Line 1 of the OUTPUT field is used to display the count. Line 2 is used to display the accumulated output value during the simulation and the average value at the end. The remaining lines can be used to display other values generated in the simulation. Line 3 in the single server simulation displays arrival time and line 4 displays the leaving time

NEW HYPERCARD FUNCTIONS

The stack contains several functions that are called by the programs used in running the various simulations:

```
function ranNum
  put ((random of 32767)/32768) into number
  return number
end ranNum

function ifThen arg1,arg2,arg3
  put arg1 into it
  if it = true then put arg2 into num else put arg3 into num
  return num
end ifThen

function sLookUp arg,table,offset,row
  put row - 1 into i
  put false into done
  repeat until done
    put i + 1 into i
    if line i + 1 of card field table is empty or ¬
    arg < word 1 of line i + 1 of card field table then
      return word offset of line i of card field table
      put true into done
    end if
  end repeat
end sLookUp
```

```
function doWalk Xo,Yo,Xn,Yn
  put false into done
  put 4 into result
  repeat until done
    put trunc(4*ranNum()) into value
    put ifThen(value < 2,2*value - 1,0) + Xo into Xo
    put ifThen(value > 1,2*(value - 2) - 1,0) + Yo into Yo
    if Xo = Xn then
      put 0 into result
    else if Yo = Yn then
      put 1 into result
    else if Xo = 0 then
      put 2 into result
    else if Yo = 0 then
      put 3 into result
    end if
    if result < 4 then
      return result
      put true into done
    end if
  end repeat
end doWalk
```

The ranNum function is similar to the @rand function in 1-2-3. No argument is needed and it generates a random number in the range from 0 to 1. The ifThen function is similar to the @if function in 1-2-3. Each function takes three argument. The first argument is a boolean expression such as leave > arrive. The value of second argument is returned if the boolean expression is true and the value of the third argument is returned if the boolean expression is false.

The sLookUp function is similar to the @vlookup in 1-2-3, but it has one additional argument, row. We created a series of lookup tables which are consecutively placed in one long table, while we created separate lookup tables in 1-2-3. The second argument identifies the long table, and the fourth argument, row, identifies the line number within the long table where a particular lookup table begins. Within the identified lookup table, the first argument identifies the value in the first column, and the third argument identifies the offset column whose value corresponding to the first argument will be returned by the sLookup function.

The doWalk function performs a procedure equivalent to the one performed by COMP in 1-2-3 shown in Figure 4. It is used to provide the control structure needed to repeat the steps in the random walk. Although it would be possible to construct the needed control structure in the program for a specific simulation, the function eliminates the need to do so. Implicitly, the doWalk function defines one set of boundaries at x and y coordinates of 0 and 0. The doWalk function takes four arguments — the x and y coordinate of the starting cell and the x and y coordinates of the other two boundaries.

SIMULATION PROGRAMS

The programs used in running a various simulations are located in a card field called BOX. The programs for the four examples are shown below:

Area Under an Exponential Curve

```
put (exp(-ranNum())) into scratch
add scratch to line 3 of card field "OUTPUT"

Note: Line 2 is the total and finally the mean of the area values.
```

Life of Two Components

```
put (-ln(1 - ranNum())) into life
put life into line 3 of card field "OUTPUT"
put (-ln(1 - ranNum())) into scratch
put scratch into line 4 of card field "OUTPUT"
put min(life,scratch) into life
add life to line 2 of card field "OUTPUT"

Note: Line 2 is the sum and finally the mean of system life.
      Line 3 is the life of first component.
      Line 4 is the life of second component.
```

Random Walk for Heat Transfer

```
put doWalk(1,1,3,3) into side
put ifThen(side = 1,100,20) into temp
put temp into line 3 of card field "OUTPUT"
add temp to line 2 of card field "OUTPUT"

Note: Line 2 is the sum and finally the mean of temperature.
```

Single Server for Advising

```
put line 3 of card field "OUTPUT" into arrive
put line 4 of card field "OUTPUT" into leave
put arrive -15*ln(ranNum()) into arrive
put arrive into line 3 of card field "OUTPUT"
put sLookUp(ranNum(),"COUNSEL",2,1) into class
put sLookUp(ranNum(),"COUNSEL",2,class) into stay
put line 2 of card field "OUTPUT" into wait
add ifThen(leave > arrive,leave - arrive,0) to wait
put wait into line 2 of card field "OUTPUT"
put  stay +  ifThen(leave > arrive,leave,arrive) into leave
put leave into line 4 of card field "OUTPUT"

Note: Line 2 is the waiting time and finally its mean
      Line 3 is  the arrival time.
      Line 4 is the leaving time.
```

REFERENCES

Oulman, Charles S. and Motoko Y. Lee, Macro Programming Using Lotus 1-2-3 for Engineers, St. Paul, MN: West Publishing Co., 1990.

Taha, Hamdy A., Operations Research, 3rd edition, New York: MacMillan Publishing Co., Inc., 1982.

```
I1: 'Monte Carlo Labeling Macro                                          READY

      I        J        K        L        M        N        O        P
1.  Monte Carlo Labeling Macro
2
3   \L        NUMBER~/rnlr~{r}/re~/wgrm~@rand~{1}{d}
4             COUNT~/rnlr~{d}
5             LIMIT~/rnlr~{r}10~{1}{d}
6             SCRATCH~/rnlr~{d}
7             TOTAL/rnlr~{d 2}
8             START~/rnlr~
9
10  Monte Carlo Computation Macro
11
12  \X        /reSCRATCH~/reTOTAL~/reCOUNT~{goto}SCRATCH~
13  LOOP      {for COUNT,1,LIMIT<1,START}
14            {let TOTAL,TOTAL/LIMIT}{calc}
15
16
17
18
19
20
```

Figure 1. Monte Carlo macros.

```
B27:                                                                     READY

      A        B        C        D        E        F        G        H
21                    Area Under an Exponential Curve
22
23  NUMBER    0.809403
24  COUNT           21
25  LIMIT           20
26  SCRATCH
27  TOTAL     0.599359
28
29  START     {calc}{let TOTAL,TOTAL+@exp(NUMBER)}
30
31
32
33
34
35
36            ------------------------------------------------------
37            {calc}{let TOTAL,TOTAL+@exp(NUMBER)}
38
39
40
```

Figure 2. Area under an exponential curve.

```
B47: (-@LN(1-NUMBER))                                                    READY

      A         B         C         D         E         F         G         H
41                      Life of Two Components
42
43  LIFE      0.623293
44  NUMBER    0.212592
45  COUNT           11
46  LIMIT           10
47  SCRATCH   0.239009
48  TOTAL     0.691545
49
50  START     {calc}(-@ln(1-NUMBER))~/rv~LIFE~
51            {calc}(-@ln(1-NUMBER))~
52            /xiLIFE>SCRATCH~/rv~LIFE~
53            {let TOTAL,TOTAL+LIFE}
54
55
56
57
58
59
60
```

Figure 3. Life of two components.

```
B68: 1                                                                   READY

      A         B         C         D         E         F         G         H
61                      Random Walk for Heat Transfer
62
63  XCOOR           1
64  YCOOR           1
65  NUMBER    0.591790
66  COUNT           11
67  LIMIT           10
68  SCRATCH          1
69  TOTAL           28
70
71  START     {let XCOOR,1}{let YCOOR,1}
72  COMP      {calc}@int(4*NUMBER)~
73            /xiSCRATCH<2~{let XCOOR,XCOOR+2*SCRATCH-1,XCOOR}
74            /xiSCRATCH>1~{let YCOOR,YCOOR+2*(SCRATCH-2)-1,YCOOR}
75            /xiXCOOR=0~{let TOTAL,TOTAL+20}/xr~
76            /xiXCOOR=3~{let TOTAL,TOTAL+20}/xr~
77            /xiYCOOR=0~{let TOTAL,TOTAL+20}/xr~
78            /xiYCOOR=3~{let TOTAL,TOTAL+100}/xr~
79            /xgCOMP~
80
```

Figure 4. Random walk for heat transfer.

```
B68: 1                                                                READY

      A        B        C        D        E        F        G        H
81                  Single Server for Advising
82
83  LAMBDA         15
84  ARR       271.6435
85  LEAVE     281.6435
86  NUMBER    0.607701
87  COUNT           21
88  LIMIT           20
89  SCRATCH
90  TOTAL     2.600419
91
92  START     {calc}(ARR-LAMBDA*@ln(NUMBER))~/rv~ARR~/re~
93            {let TOTAL,@if(LEAVE<ARR,TOTAL,TOTAL+LEAVE-ARR)}
94  COND2     {calc}@vlookup<NUMBER,CLASS,1)~/rv~TABLE1~/rv~TABLE2~/re~
95  COND3     {calc}@if(LEAVE<ARR,ARR+@vlookup<NUMBER,
96  TABLE1    FRESHMAN
97            ,1),LEAVE+@vlookup(NUMBER,
98  TABLE2    FRESHMAN
99            ,1))~/rv~LEAVE~/re~
100
```

Figure 5. Single server for advising.

```
B109:                                                                 READY

      A        B        C        D        E        F        G        H
101
102                         Class
103                   Prob Level
104                      0 FRESHMAN
105                   0.35 SOPHOMORE
106                   0.65 JUNIOR
107                   0.85 SENIOR
108
109                         Freshman          Sophomore
110                   Prob    Minutes     Prob    Minutes
111           Add/Drop   0          2        0          3
112           Advice   0.1         17      0.2         20
113           Schedule 0.3         10      0.3          8
114
115                           Junior            Senior
116           Add/Drop   0          4        0          5
117           Advice   0.3         23      0.4         26
118           Schedule 0.5          6      0.6          8
119
120
```

Figure 6. Lookup tables for single server in Lotus 1-2-3.

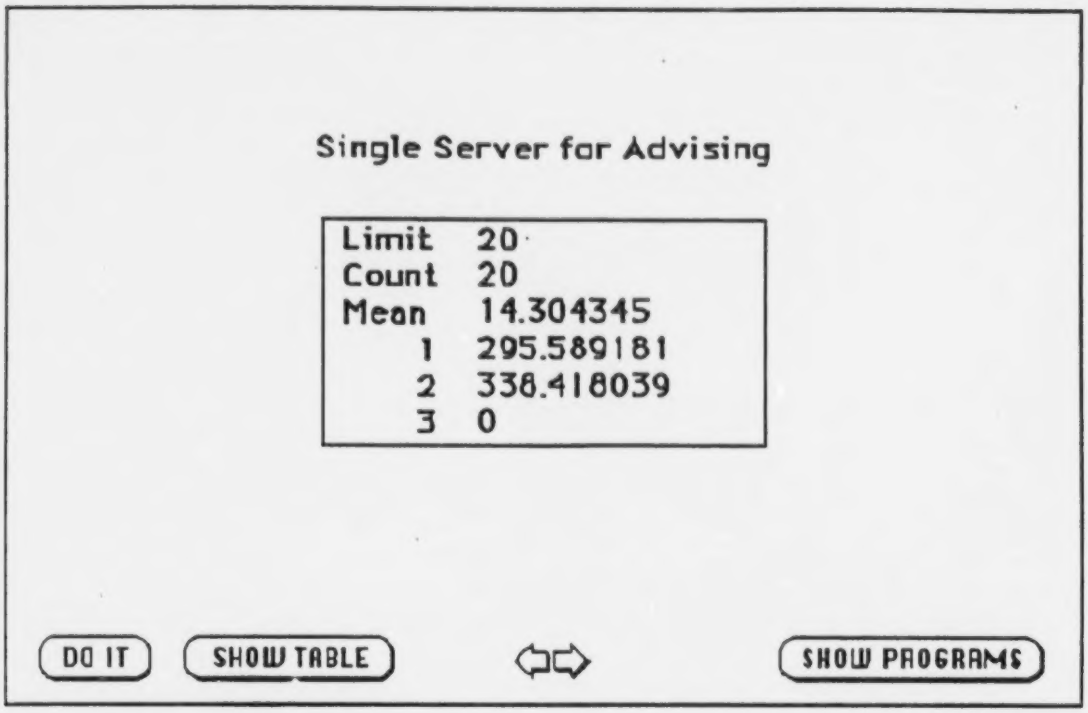

Figure 7. Card for single server.

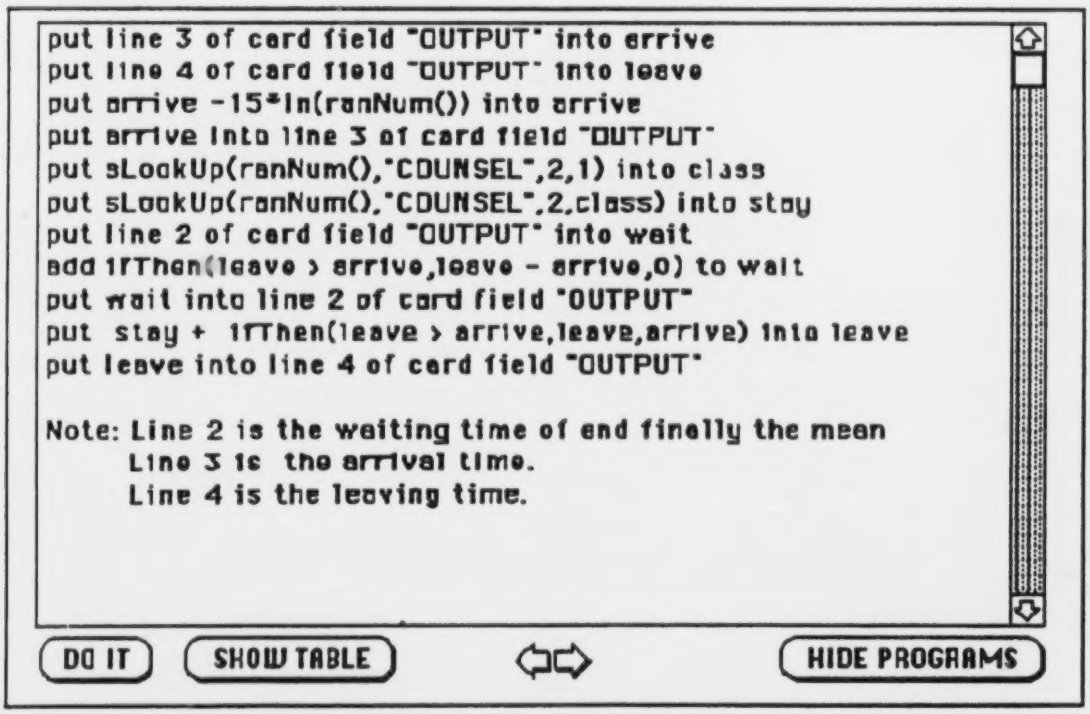
put line 3 of card field "OUTPUT" into arrive
put line 4 of card field "OUTPUT" into leave
put arrive -15*ln(ranNum()) into arrive
put arrive into line 3 of card field "OUTPUT"
put sLookUp(ranNum(),"COUNSEL",2,1) into class
put sLookUp(ranNum(),"COUNSEL",2,class) into stay
put line 2 of card field "OUTPUT" into wait
add ifThen(leave > arrive,leave - arrive,0) to wait
put wait into line 2 of card field "OUTPUT"
put stay + ifThen(leave > arrive,leave,arrive) into leave
put leave into line 4 of card field "OUTPUT"
Note: Line 2 is the waiting time of end finally the mean
Line 3 is the arrival time.
Line 4 is the leaving time.
DO IT
SHOW TABLE
HIDE PROGRAMS

Figure 8. Program for single server.

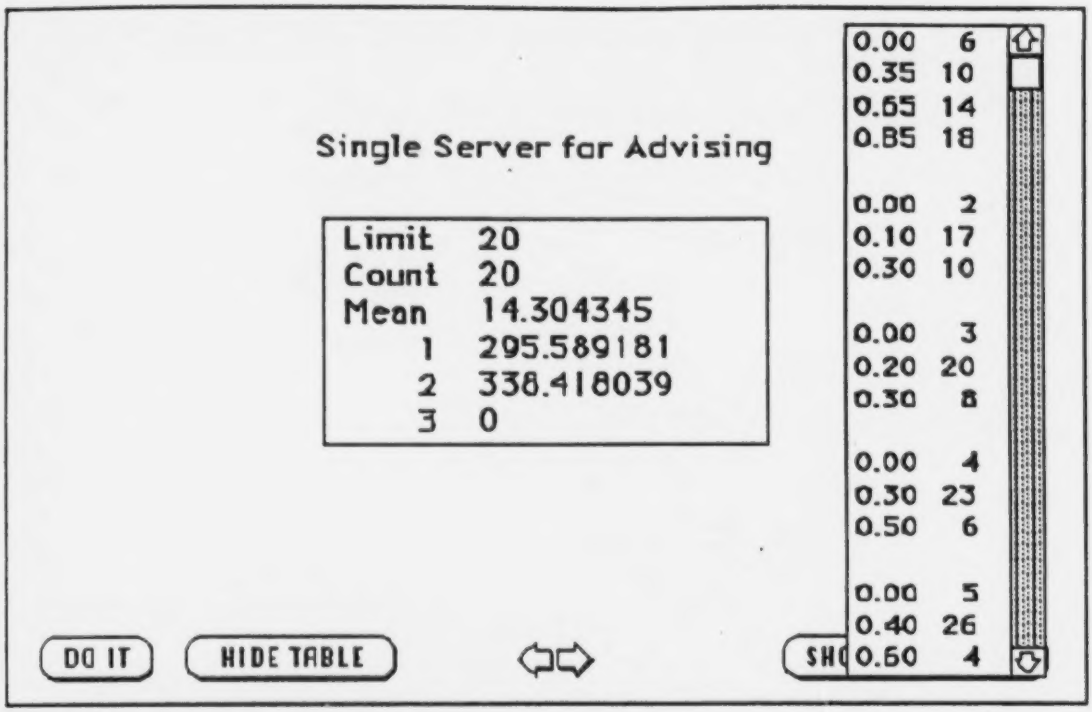

Figure 9. Lookup table for single server using HyperCard.

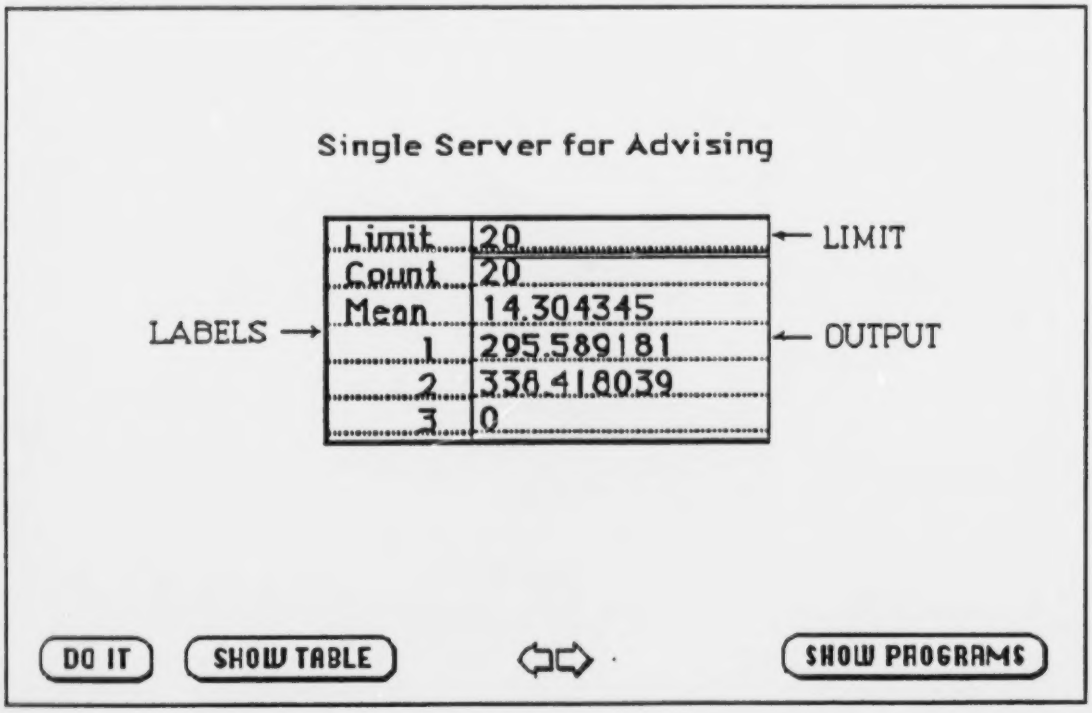

Figure 10. Fields used in single server.